Analisis y Diseño de Algoritmos

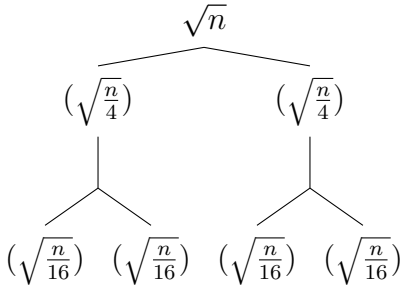
Juan Miguel Rojas

19 de febrero de 2024

1. Ejercicio 1.6 (A, E, I): Recursion Trees (Erickson, página 49).

Punto A.

$A(n) = 2 \cdot E(\frac{n}{3}) + n$



Con esto, podemos denotar que la suma de cada nivel es igual a $\sqrt{n}$, como por ejemplo tomaremos el nivel 2, y haremos la suma:

$A(\frac{n}{16}) = (\sqrt{\frac{n}{16}}) + (\sqrt{\frac{n}{16}}) + (\sqrt{\frac{n}{16}}) + (\sqrt{\frac{n}{16}})$

$A(\frac{n}{16}) = \sqrt{n}$

Luego de darnos cuenta de que es correcta la suma y nos da el resultado esperado, procedo a hallar la altura del arbol.

Altura

Igualamos a una costante para despejar tratar de despejar i, en este caso utilizaremos 1, porqué es la mas sencilla.

$\frac{n}{4^i} = 1$

$n = 4^i$

$i = \log_4(n)$

Costo por niveles

Primero para saber el numero de nodos de un nivel es 2^i y para hallar los nodos de un nivel sería:

$2^i \cdot (\sqrt{\frac{n}{4^i}}) = \cancel{2^i} \cdot (\frac{\sqrt{n}}{\cancel{2^i}}) = \sqrt{n}$

Complejidad

Ahora, hallaremos la complejidad con notacion O.

$$A(n) = O(\sum_{i=0}^{\log_4(n)} \sqrt{n})$$

$$A(n) = O(\sqrt{n} \cdot \sum_{i=0}^{\log_4(n)} 1)$$

$$A(n) = O(\sqrt{n} \cdot (1 + 1 + 1 + 1 + ... + 1))$$

Ahora, sabemos que el limite de la sumatoria es $\log_4(n)$, por lo tanto $(1+1+1+1+...+1)$ es igual a $\log_4(n) + 1$, porque va desde 0 hasta ese valor, por eso el $+1$.

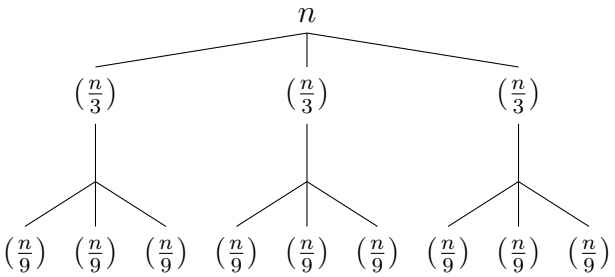
$$A(n) = O(\sqrt{n} \cdot \log_4(n) + 1)$$

La constante de igual, entonces la quitamos y lo mismo con la base del logaritmo.

$$A(n) = O(\sqrt{n} \cdot \log(n))$$

Punto E.

$E(n) = 3 \cdot E(\frac{n}{3}) + n$



Con esto, podemos denotar que la suma de cada nivel es igual a n, como por ejemplo tomaremos el nivel 2, y haremos la suma:

$E(\frac{n}{3}) = (\frac{n}{3}) + (\frac{n}{3}) + (\frac{n}{3}) + (\frac{n}{3})$

$E(\frac{n}{3}) = n$

Luego de darnos cuenta de que es correcta la suma y nos da el resultado esperado, procedo a hallar la altura del arbol.

Altura

Igualamos a una costante para despejar tratar de despejar i, en este caso utilizaremos 1, porqué es la mas sencilla.

$\frac{n}{3^i} = 1$

$n = 3^i$

$i = \log_3(n)$

Costo por niveles

Primero para saber el numero de nodos de un nivel es 3^i y para hallar los nodos de un nivel sería:

$3^i \cdot (\frac{n}{3^i}) = \cancel{3^i} \cdot (\frac{n}{\cancel{3^i}}) = n$

Complejidad

Ahora, hallaremos la complejidad con notacion O.

$$E(n) = O(\sum_{i=0}^{\log_3(n)} n)$$

$$E(n) = O(n \cdot \sum_{i=0}^{\log_3(n)} 1)$$

$$E(n) = O(n \cdot (1 + 1 + 1 + 1 + ... + 1))$$

Ahora, sabemos que el limite de la sumatoria es $\log_3(n)$, por lo tanto $(1+1+1+1+...+1)$ es igual a $\log_3(n) + 1$, porque va desde 0 hasta ese valor, por eso el $+1$.

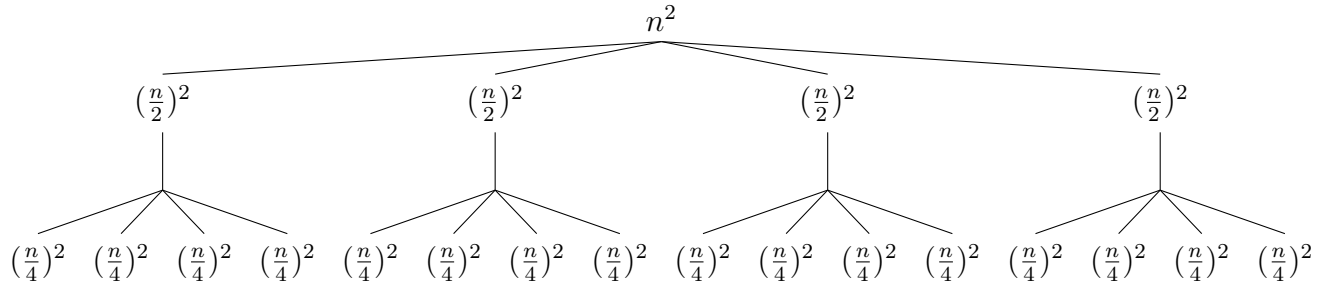
$$E(n) = O(n \cdot \log_3(n) + 1)$$

La constante de igual, entonces la quitamos y lo mismo con la base del logaritmo.

$$E(n) = O(n \cdot \log(n))$$

Punto I.

$I(n) = 4 \cdot I(\frac{n}{2}) + n^2$



Con esto, podemos denotar que la suma de cada nivel es igual a n^2, como por ejemplo tomaremos el nivel 2, y haremos la suma:

$I(\frac{n}{2}) = (\frac{n}{2})^2 + (\frac{n}{2})^2 + (\frac{n}{2})^2 + (\frac{n}{2})^2$

$I(\frac{n}{2}) = n^2$

Luego de darnos cuenta de que es correcta la suma y nos da el resultado esperado, procedo a hallar la altura del arbol.

Altura

Igualamos a una costante para despejar tratar de despejar i, en este caso utilizaremos 1, porqué es la mas sencilla.

$\frac{n}{2^i} = 1$

$n = 2^i$

$i = \log_2(n)$

Costo por niveles

Primero para saber el numero de nodos de un nivel es 4^i y para hallar los nodos de un nivel sería:

$4^i \cdot (\frac{n}{2^i})^2 = \cancel{4^i} \cdot (\frac{n^2}{\cancel{4^i}}) = n^2$

Complejidad

Ahora, hallaremos la complejidad con notacion O.

$$I(n) = O(\sum_{i=0}^{\log_2(n)} n^2)$$

$$I(n) = O(n^2 \cdot \sum_{i=0}^{\log_2(n)} 1)$$

$$I(n) = O(n^2 \cdot (1 + 1 + 1 + 1 + ... + 1))$$

Ahora, sabemos que el limite de la sumatoria es $\log_2(n)$, por lo tanto $(1+1+1+1+...+1)$ es igual a $\log_2(n) + 1$, porque va desde 0 hasta ese valor, por eso el $+1$.

$$I(n) = O(n^2 \cdot \log_2(n) + 1)$$

La constante de igual, entonces la quitamos y lo mismo con la base del logaritmo.

$$I(n) = O(n^2 \cdot \log(n))$$

2. Ejercicio 1.16: Weighted Median (Erickson, página 52).

Suppose we are given a set S of n items, each with a value and a weight. For any element $x \in S$, we define two subsets

- $S < x$ is the set of elements of S whose value is less than the value of x.
- $S > x$ is the set of elements of S whose value is more than the value of x.

For any subset $R \subseteq S$, let $w(R)$ denote the sum of the weights of elements in R. The weighted median of R is any element x such that $w(S < x) \leq w(S)/2$ and $w(S > x) \leq w(S)/2$.
Describe and analyze an algorithm to compute the weighted median of a given weighted set in O(n) time. Your input consists of two unsorted arrays $S[1..n]$ and$W[1..n]$, where for each index i, the ith element has value $S[i]$ and weight $W[i]$. You may assume that all values are distinct and all weights are positive.

Solución

Para solucionar este problema, tomaremos algo de dividir y conquistar. Primero tenemos un conjunto de S de tamaño n y un conjunto W de tamaño n también el cual contendrá los pesos, el cual se partira en dos subconjuntos l y r (izquierda y derecha) los cuales surgen a partir de un x siendo $l < x$ para cada elemento de l y $r > x$ para cada elemento de r.
Primeramente, el problema tenemos que tomar un pivote el cual será nuestro x, este se selecciona de manera aleatoria entre 0 y $n-1$, el cual serían las posiciones de nuestro conjunto. Una vez obtenido nuestro pivote, partimos las dos listas dependiendo del valor del pivote, y luego verificamos si la suma de los dos subconjuntos son menores o iguales que el pivote y si se cumple, devuelve el x.
En el caso de que con el x obtengamos un subconjunto el cual su suma es mayor que l ó r tendremos que recurrir y nuestro nuevo x sera (denotare len como una funcion con el tamaño del conjunto y m como el subconjunto que su suma es mayor que $w(S)/2)$ $len(m) - 1//2$ y este será nuestro nuevo x. Cabe resaltar que en l y r, no guardaremos los valores, sino que la posición del valor en S, es decir, si tengo un valor 4 en S que pertenece a l, no guardare el 4, sino que, la posición del 4 en S.

Pseudocodigo

Nuestro pseudocodigo sería tal que así.

Algorithm 1 Calcular un x el cual obtenga dos subconjuntos tal que su suma sea menor que $w(S)/2$

```
solve(S, x)
i, ans ← 0
f ← longitud(A) − 1
l ← obtenerL(S, x)
r ← obtenerR(S, x)
sumaL ← suma(l)
sumaR ← suma(r)
if sumaL y sumaR son menores o iguales que w(S)/2 then
    ans ← x
else
    if sumaL es mayor que w(S)/2 then
        x ← (longitud(l) − 1)//2
        ans ← solve(A, x)
    else
        x ← (longitud(r) − 1)//2
        ans ← solve(A, x)
    end if
end if
```

3. Ejercicio 1.38: Central Vertices (Erickson, página 64).

Let T be a binary tree with n vertices. Deleting any vertex v splits T into at most three subtrees, containing the left child of v (if any), the right child of v (if any), and the parent of v (if any). We call v a central vertex if each of these smaller trees has at most

n/2 vertices. See Figure 1.27 for an example. Describe and analyze an algorithm to find a central vertex in an arbitrary given binary tree. [Hint: First prove that every tree has a central vertex.]

Solución

Para solucionar este problema partire con algo parecido a la solución del anterior punto que es la aleatoriedad. Seleccionaremos un vertice al azar, el cual eliminaremos y hallaremos la suma de los subarboles restantes. Y si estas tienen como suma maximo n/2, entonces este será nuestro vertice central, pero en caso que no, tendremos al menos un subarbol mayor que n/2 y tendremos que aplicar esta misma estrategia para este sub arbol hasta encontrar el vertice que cumpla la condición.

Correctitud

El problema que se evitara es que aleatoriamente pueda caer en el mismo vertice, pero este se soluciona ya que se navega entre los subarboles, no dentro del mismo, así que no navegaremos en un mismo vertice.

Complejidad

La complejidad sería en el peor de los casos $O(n)$, ya que tendría que tomar todos los vertices para buscar el vertice central.